TRIGGER

LABORATÓRIO DE BANCO DE DADOS

VANESSA BORGES

Definição de gatilhos
TRIGGER

- **Um gatilho é um comando que é executado pelo sistema automaticamente, em consequência de uma modificação no banco de dados**

- Duas exigências devem ser satisfeitas para a projeção de uma mecanismo de gatilho:
  - Especificar a condição sob as quais o gatilho deve ser executado
  - Especificar as ações que serão tomadas quando um gatilho for disparado

Definição de gatilhos
TRIGGER

- Bloco PL/SQL que é disparado de forma automática e implícita sempre que ocorrer um evento associado a uma tabela
  - INSERT
  - UPDATE
  - DELETE
  - TRUNCATE

- Não pode ser chamado explicitamente

Gatilhos – quando é utilizado

- Manutenção de tabelas de auditoria
- Implementação de níveis de segurança mais complexos
- Geração de valores de colunas referentes a atributos derivados
- Validação de restrições de integridade mais complexas que as suportadas diretamente pelo SGBD
- Mecanismos de alerta para iniciar ações quando certas condições são satisfeitas
  - Exemplo: suponha que um depósito queira manter um estoque mínimo de determinado produto. Se o nível for inferior ao mínimo uma ação pode ser disparada

Definição de gatilhos
CREATE TRIGGER

- Sintaxe padrão da SQL que a maioria dos bancos de dados implementam versões fora desse padrão
- **Sintaxe:**

```
CREATE TRIGGER <nome_do_gatilho>
{AFTER | BEFORE | INSTEAD OF}
{event [OR ...]}
ON <nome_tabela>
FOR EACH {ROW | STATEMENT}
[WHEN (<condição boleana>)]
EXECUTE [FUNCTION | PROCEDURE]
<nome da função> (<lista de parâmetros>);
```

When	Event	Row-level	Statement-level
BEFORE (antes do evento)	INSERT/UPDATE/ DELETE	Tables and foreign key tables	Tables, views and foreign key tables
	TRUNCATE	—	Tables
AFTER (depois do evento)	INSERT/UPDATE/ DELETE	Tables and foreign key tables	Tables, views and foreign key tables
	TRUNCATE	—	Tables
INSTEAD OF (ao invés de)	INSERT/UPDATE/ DELETE	Views	—
	TRUNCATE	—	—

https://www.postgresql.org/docs/current/sql-createtrigger.html

Definição de gatilhos
CREATE TRIGGER

- Sintaxe padrão da SQL que a maioria dos bancos de dados implementam versões fora desse padrão
- **Sintaxe:**

```
CREATE TRIGGER <nome_do_gatilho>
{AFTER | BEFORE | INSTEAD OF}
{event [OR ...]}
ON <nome_tabela>
FOR EACH {ROW | STATEMENT}
[WHEN (<condição boleana>)]
EXECUTE [FUNCTION | PROCEDURE]
<nome da função> (<lista de parâmetros>);
```

- **Eventos** de disparo: o gatilho somente será disparado nessas condições
  - UPDATE [OF <lista de atributos>]
  - DELETE
  - INSERT
  - TRUNCATE

Definição de gatilhos
CREATE TRIGGER

- Sintaxe padrão da SQL que a maioria dos bancos de dados implementam versões fora desse padrão
- **Sintaxe:**

```
CREATE TRIGGER <nome_do_gatilho>
{AFTER | BEFORE | INSTEAD OF}
{event [OR ...]}
ON <nome_tabela>
FOR EACH {ROW | STATEMENT}
[WHEN (<condição boleana>)]
EXECUTE [FUNCTION | PROCEDURE]
<nome da função> ();
```

- ROW: nível de linha
- STATEMENT: nível de instrução

- Por exemplo, se uma instrução UPDATE for executada, e esta afetar seis linhas, temos que a trigger de nível de linha será executada seis vezes, enquanto que a trigger a nível de instrução será chamada apenas uma vez por instrução SQL

When	Event	Row-level	Statement-level
BEFORE (antes do evento)	INSERT/UPDATE/ DELETE	Tables and foreign key tables	Tables, views and foreign key tables
	TRUNCATE	—	Tables
AFTER (depois do evento)	INSERT/UPDATE/ DELETE	Tables and foreign key tables	Tables, views and foreign key tables
	TRUNCATE	—	Tables
INSTEAD OF	INSERT/UPDATE/ DELETE	Views	—
	TRUNCATE	—	—

Definição de gatilhos
Elementos

- Um gatilho pode acessar elementos da tupla modificada usando as variáveis **NEW.**<nome-columa> e **OLD.**<nome-coluna>
  - Sempre vinculados à tabela desencadeadora do gatilho
  - Onde:
    - NEW referencia o valor atual
    - OLD o valor anterior à modificação, e o campo é indicado por nome-coluna.

instrução / identificador	`old`	`new`
`INSERT`	NULL	valores que serão inseridos
`UPDATE`	valores antes da atualização	novos valores para a atualização
`DELETE`	valores antes da remoção	NULL

Exemplo - UPDATE
Valida atualização de saldo

```
-- Definição da function apenas atualiza o dados se o novo saldo for diferente
do anterior
CREATE OR REPLACE FUNCTION valida_alteracao_conta() RETURNS TRIGGER AS $$
BEGIN
    if old.saldo is distinct from new.saldo then
        RAISE NOTICE 'Atualizado old: %, new: %', old.saldo, new.saldo;
        return new;
    end if;
    return null;
END;
$$ LANGUAGE PLPGSQL;
```

Exemplo - UPDATE
Valida atualização de saldo

```
-- Executa a function antes que a tabela conta seja atualizada
CREATE TRIGGER valida_alteracao
    BEFORE UPDATE ON conta FOR EACH ROW
    EXECUTE FUNCTION valida_alteracao_conta();

-- Executa a function antes que o atributo saldo da tabela conta seja atualizado
CREATE TRIGGER valida_alteracao
    BEFORE UPDATE OF saldo ON conta FOR EACH ROW
    EXECUTE FUNCTION valida_alteracao_conta();

-- Executa a function sempre que houver alteração na tabela conta e o saldo
-- tiver sido de fato alterado (condição when).
CREATE TRIGGER valida_alteracao
    BEFORE UPDATE ON conta FOR EACH ROW
    WHEN (OLD.saldo IS DISTINCT FROM NEW.saldo)
    EXECUTE FUNCTION valida_alteracao_conta();
```

Habilitar e Desabilitar um gatilho

- Habilitar um gatilho

```
ALTER TABLE <nome da tabela> ENABLE TRIGGER <nome do trigger>;
```

- Habilitar todos os gatilhos

```
ALTER TABLE <nome da tabela> ENABLE TRIGGER ALL;
```

- Desabilitar um gatilho

```
ALTER TABLE <nome da tabela> DISABLE TRIGGER <nome do trigger>;
```

- Desabilitar todos os gatilho

```
ALTER TABLE <nome da tabela> DISABLE TRIGGER ALL;
```

Alterar um gatilho

- Alterar um gatilho
  - Muda a definição de um gatilho

```
ALTER TRIGGER <nome do gatilho>
ON <nome da tabela> RENAME TO <novo nome do gatilho> ;
```

Apagar um gatilho

- Remove um gatilho

```
DROP TRIGGER [ IF EXISTS ] <nome do gatilho>
    ON <nome da tabela> [ CASCADE | RESTRICT ];
```

Onde:

- CASCADE: automaticamente elimine objetos que dependem do gatilho
- RESTRICT: Rejeita a operação de remoção se algum objeto depender do gatilho. Este é o padrão.

Variáveis especiais - plpgsql

- VARIÁVEIS ESPECIAIS são criadas automaticamente
  - NEW: tipo RECORD contendo nova linha em INSERT/UPDATE
  - OLD: tipo RECORD contendo linha antiga em UPDATE/DELETE
  - TG_NAME: o nome do gatilho
  - TG_WHEN: BEFORE or AFTER
  - TG_LEVEL: ROW or STATEMENT
  - TG_OP: INSERT, UPDATE or DELETE
  - TG_RELID: object ID da tabela
  - TG_RELNAME: nome da tabela
  - TG_NARGS: número de argumentos
  - TG_ARGV[]: array contendo argumento

Variáveis especiais: TG_OP

- O PostgreSQL guarda em memória algumas variáveis que identificam que tipo de ação que será realizada
  - **TG_OP** pode possuir os valores: ’INSERT’, ’UPDATE’, ’DELETE’

- Exemplo:

```
CREATE OR REPLACE FUNCTION testeAcao() RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP='INSERT' THEN
    ELSIF TG_OP='DELETE' THEN
    ELSIF TG_OP='UPDATE' THEN
    ENF IF;
    RETURN NEW;
END;
$$ LANGUAGE PLPGSQL;
```

Definição de gatilhos - RESUMO

- O *trigger* pode ser especificada para executar antes (BEFORE) ou depois (AFTER) das operações de inserção (INSERT), remoção (DELETE) e atualização (UPDATE).
- Se um *trigger* for marcado com os comandos FOR EACH ROW, é invocada uma vez para cada tupla que a operação afetar. Se o *trigger* é marcado com o comando FOR EACH STATEMENT, é executada somente uma vez para cada operação, não importando quantas tuplas a operação afete.
- O *trigger* pode acessar elementos da tupla modificada usando as variáveis NEW.nome-columa e OLD.nome-coluna
  - Onde NEW referencia o valor atual e OLD o valor anterior à modificação, e o campo é indicado por nome-coluna
- Se forem fornecidos vários *triggers* para o mesmo evento, eles serão executados em ordem alfabética
- Os *triggers* são automaticamente removidos se a tabela a qual estiverem associadas for removida
- Funções de Trigger precisam obrigatoriamente de algum retorno.